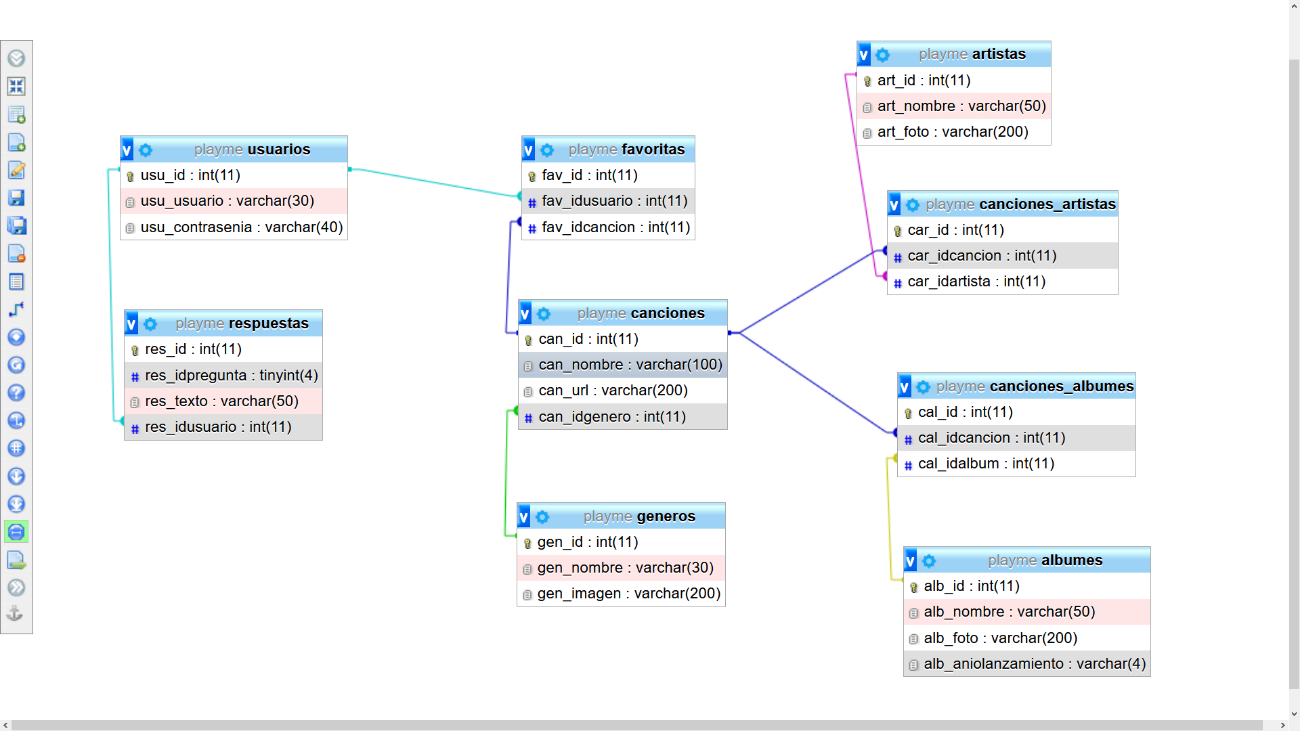
Base de datos del proyecto PlayMe

La base de datos que utilizo en mi proyecto PlayMe, que como comenté en el documento anterior, es un reproductor de música online, es SQL, utilizando María DB, que es el motor de bases de datos que implementa XAMPP en la versión v7.4.6 que utilizo para virtualizar el servidor de manera local. Aunque próximamente lo estaré ejecutando en el espacio del servidor que el profesor Daián nos facilitó.

La finalidad de este texto es explicar la estructura de la base de datos que desarrollé e implementé en mi proyecto, explicando y presentando el material necesario para justificar las decisiones que me llevaron a dicha estructura.

Diagrama de la estructura de la base de datos



En principio voy a empezar por el usuario, el cual se registra en el sitio con la finalidad de asociar un perfil del mismo a las selecciones que realice en cuanto a canciones favoritas, las cuales, serán utilizadas más tarde por el sistema para sugerirle listas de reproducción que puedan ser del agrado del usuario basándome justamente en las canciones que vaya marcando como favoritas. La idea es tomar el género de las canciones que el usuario marque como favoritas y armar listas de reproducción separadas por dichos géneros, por lo tanto si marca canciones favoritas que correspondan al género Rock y Blues, tendrá 2 listas de reproducción separadas por dichos géneros y dichas listas estarán basadas en las canciones favoritas que vayan marcando todos los usuarios que interactúen con el sitio web; por lo tanto a mayor cantidad de usuarios vayan utilizando el sitio, más canciones irán engrosando estas listas de favoritos que servirán para ser sugeridas a los usuarios que les agrade dicho género, obviamente con un límite de cantidad que actualmente lo tengo funcionando en 10 canciones como más adelante mostraré en la correspondiente consulta SQL que ejecuto para obtener dichos datos.

La tabla usuarios contiene el nombre de usuario y su contraseña, la idea es realizar un sitio para poder escuchar música sin vender suscripciones a los mismos ni productos, por lo tanto solo solicito esa información al registrarse para que el sitio no se sienta invasivo, como tantos otros sitios que solicitan el correo electrónico u otra información para luego enviarnos publicidad molesta o recomendaciones de planes de suscripción que no deseamos recibir, o simplemente para vender nuestros datos a otras empresas de diversos rubros. Obviamente el usuario y la contraseña deben cumplir ciertos requisitos en cuanto a su formato que son informados al usuario al ingresar dichos datos y validados por el sistema para que se cumpla dicho formato.

Asociada a esta tabla se encuentra una tabla que contiene respuestas a preguntas de recuperación de contraseña que el usuario debió ingresar al registrarse. En esta tabla guardo el ID del usuario, el Id de la respuesta y la respuesta a la pregunta correspondiente. Los ID de las preguntas se encuentran dentro del mismo código del sistema, dado que son solo 7 preguntas, de las cuales el usuario está obligado a llenar 3 de las mismas para ser guardadas en la base de datos como opciones de recuperación de contraseña en caso de olvidarse de la misma. Por lo tanto, sabiendo el usuario y el Id de la pregunta que seleccione para recuperar su contraseña en caso de olvido, solo tengo que comparar la respuesta que ingresó con la almacenada en la tabla para luego de ser validada correctamente dicha respuesta de recuperación, llevarlo a una interfaz de ingreso de una nueva contraseña.

Luego tenemos la tabla favoritas que es donde se van registrando las canciones marcadas como favoritas por parte del usuario, la misma es una tabla mediadora entre usuarios y canciones dado que un usuario puede tener varias canciones favoritas y una canción puede ser elegida como favorita por varios usuarios.

La tabla canciones incluye una llave foránea correspondiente al ID del género musical. Asocié el género a la canción y no al álbum dado que en un álbum puede haber canciones de distintos géneros, como en el caso de álbumes en donde colaboran diversos artistas.

La vinculación entre las tablas canciones y artistas se realiza mediante una tabla mediadora llamada canciones\_artitstas, dado que un artista tiene varias canciones y una canción puede pertenecer a varios artistas como en el caso de artistas que trabajan de manera colaborativa en una determinada canción, como a veces ocurre con las que forman parte de la banda de sonido de una película o los álbumes que realizan algunos artistas que invitan a otros músicos para realizar un determinado proyecto musical.

Por otro lado tenemos la tabla canciones\_albumes, la cual vincula las canciones con los álbumes, esta tabla mediadora la conformé dado que un álbum contiene varias canciones y una misma canción puede estar en varios álbumes como es el caso de las canciones que se incluyen en las antologías o en los “Best of”, la misma canción se encuentra en el álbum original de lanzamiento y luego puede ser incluida en una recopilación para formar parte de una antología.

Mediante esta estructuración de la base de datos puedo acceder a los datos necesarios para los distintos procedimientos que lleva adelante el sistema como por ejemplo los diversos filtros que se pueden aplicar.

Filtro por canciones (código java script):

let consulta = "SELECT alb\_id, alb\_nombre, alb\_foto, art\_nombre" +

" FROM albumes" +

" INNER JOIN canciones\_albumes" +

" ON albumes.alb\_id = canciones\_albumes.cal\_idalbum" +

" INNER JOIN canciones" +

" ON canciones\_albumes.cal\_idcancion = canciones.can\_id" +

" INNER JOIN canciones\_artistas" +

" ON canciones.can\_id = canciones\_artistas.car\_idcancion" +

" INNER JOIN artistas" +

" ON canciones\_artistas.car\_idartista = artistas.art\_id" +

" WHERE can\_nombre LIKE \"%" + texto\_filtro + "%\"" +

" GROUP BY alb\_id;";

En esta consulta filtro las tablas por el nombre de la canción, la cual debe contener el texto ingresado por el usuario en algún lugar de la cadena que conforma el nombre de la canción, esto se logra utilizando la sentencia LIKE con los comodines % al principio y final de la cadena de búsqueda contendida en el LIKE. Se puede observar una barra inclinada previa a las comillas que abren y cierran la sentencia LIKE, esto es para evitar el problema que puede surgir si se introduce un nombre de canción para la búsqueda que contenga apóstrofes en el texto, los cuales son comunes en nombres de canciones en inglés; por ejemplo “Rock ‘n’ roll” o “I don’t want money”. Dado que las comillas dobles ya las utilizaba para la cadena de la consulta, y siendo que la búsqueda que se incluye en la sentencia LIKE debe ir entrecomillada, y las mismas no podían ser comillas simples porque se confundirían con las mismas que conforman el nombre de la canción a buscar, ser resuelve incorporando la barra que indica que no se interpreten las comillas que siguen a la barra como comillas de cierre de cadena, sino que sean pasadas como parte de la cadena de la consulta. Así logré que las búsquedas de las canciones con nombres que contengan comillas simples, funcionen correctamente.

Mediante los INNER JOIN incluidos en la consulta formo los puentes necesarios entre las tablas para recopilar los datos que necesito para los distintos procesos o para ser mostrados al usuario. El signo más que se encuentra al final de las líneas es solo para concatenar las distintas líneas de la consulta; JavaScript no permite escritura multilínea sino es concatenando las cadenas y por una cuestión de prolijidad sintáctica del código, prefiero escribirlas en varias líneas, lo cual simplifica su lectura y corrección.

El motivo por el cual escribo algunas consultas a la base de datos en JavaScript, la de los diferentes filtros, es porque para procesar dichos filtros utilizo un archivo php al cual le paso como parámetro la consulta y proceso la respuesta devuelta por el mismo en JavaScript. Esto me permitió utilizar un único archivo php, el cual me devuelve un array asociativo, como se denomina a los array generados por la sentencia “mysqli\_fetch\_all($resultado, MYSQLI\_ASSOC);”, la cual tiene como ventaja el devolver los datos en el siguiente formato “{“alb\_id”:”1”,”alb\_nombre”:”The besto of rock”}”, lo cual simplifica el procesamiento de los datos devueltos por el servidor; es como el formato llave valor de otros lenguajes como Python, en donde dicho formato se lo denomina diccionario, que asocian un valor a una llave que los identifica y dado que en este caso la clave llave es el nombre del campo, ya se que nombres tendrán al recibirlos, dado que serán los mismos que solicité al servidor en la consulta que pasé como parámetro; de ahí el nombre de asociativo, los datos vuelven al cliente asociados a una llave que en este caso es el nombre de los campos de las tablas consultadas.

En el filtro por nombre de canción utilizo la combinación de WHERE y LIKE para el filtrado y uso la sentencia GROUP BY alb\_id para traer el álbum que contiene dicha canción y las demás canciones que lo integran, de este modo muestro las canciones de dicho álbum pero resaltando en otro color la canción que contiene el texto ingresado en la búsqueda.

Código para el envío de la consulta y recepción de la respuesta del servidor

//Código js

$.ajax ({

url: "../php/cargaAlbumes.php",

type: "POST",

async: false,

data: {cadena:\_consulta}, //consulta contiene la cadena que procesará php.

success: function(respuesta) {

JSON.parse(respuesta, function(clave, valor) {

if (clave == "alb\_id") {

id\_album.push(valor);

}

if (clave == "alb\_nombre") {

nombre\_album.push(valor);

}

if (clave == "alb\_foto") {

foto\_album.push(valor);

}

if (clave == "art\_nombre") {

artista\_album.push(valor);

}

});

}

});

};

Código del archivo php que procesa las consulta de los filtros

<?php

//Módulo para la búsqueda de los datos.

$consulta = $\_POST["cadena"];

//Traemos la conexión.

include("conexion.php");

if ($cnx) {

$resultado = mysqli\_query($cnx, $consulta);

$datos = mysqli\_fetch\_all($resultado, MYSQLI\_ASSOC);

}else{

die("Fallo en la conexión");

}

mysqli\_close($cnx);

echo json\_encode($datos);

?>

Consulta para obtener la lista de sugerencias de un determinado género

"SELECT count(\*) AS cantidad, can\_id, can\_nombre,can\_url, art\_nombre, art\_foto" +

" FROM favoritas" +

" INNER JOIN canciones" +

" ON favoritas.fav\_idcancion = canciones.can\_id" +

" INNER JOIN canciones\_artistas" +

" ON canciones.can\_id = canciones\_artistas.car\_idcancion" +

" INNER JOIN artistas" +

" ON canciones\_artistas.car\_idartista = artistas.art\_id" +

" WHERE can\_idgenero = " + String(\_id\_genero\_seleccionado) +

" GROUP BY can\_id" +

" ORDER BY cantidad DESC" +

" LIMIT 10;";

Mediante la consulta SQL arriba expuesta obtengo los datos de las canciones que formarán las listas de sugeridos para el usuario. En la misma agrupo y cuento las canciones qué más veces fueron marcadas como favoritas por los usuarios. La sentencia WHERE selecciona las canciones cuyo ID de género musical sea igual al que otro proceso del sistema determinó como favorito del usuario y la función String de js convierte el ID de numérico a cadena para ser integrado en la cadena de la consulta. GROPU BY agrupa las canciones por el id de las canciones, si no las agrupara, la consulta me devolvería un solo valor correspondiente a la cantidad de las canciones que cumplieran con la condición del WHERE. Luego con ORDER BY obtengo una lista ordenada por la cantidad de veces que fue marcada como favorita dentro de la tabla favoritas por otros usuarios, la misma la ordeno de manera descendente para poder obtener las cantidades más grandes al principio de la lista, es decir, las canciones que más veces hayan sido marcadas como favoritas aparecerán al principio de dicha lista, y finalmente limito la lista a 10 canciones.

Diagrama entidad relación

